Reprinted from
Journal of Imaging Science and Technology® 61(1): 010405-1–010405-13, 2017.

Multi-GPU Acceleration of Branchless Distance Driven Projection and Backprojection for Clinical Helical CT

Ayan Mitra
Department of Electrical and Systems Engineering, Washington University, 1 Brookings Drive, Saint Louis, MO, 63130
E-mail: a.mitra@wustl.edu

David G. Politte
Mallinckrodt Institute of Radiology, Washington University School of Medicine, 510 South Kingshighway Blvd, Saint Louis, MO, 63110

Bruce R. Whiting
Department of Radiology, University of Pittsburgh, Pittsburgh, PA 15213

Jeffrey F. Williamson
Department of Radiation Oncology, Virginia Commonwealth University, Richmond, VA 23298

Joseph A. O'Sullivan▲
Department of Electrical and Systems Engineering, Washington University, 1 Brookings Drive, Saint Louis, MO, 63130

Abstract. *Model-based image reconstruction (MBIR) techniques have the potential to generate high quality images from noisy measurements and a small number of projections which can reduce the x-ray dose in patients. These MBIR techniques rely on projection and backprojection to refine an image estimate. One of the widely used projectors for these modern MBIR based technique is called branchless distance driven (DD) projection and backprojection. While this method produces superior quality images, the computational cost of iterative updates keeps it from being ubiquitous in clinical applications. In this paper, we provide several new parallelization ideas for concurrent execution of the DD projectors in multi-GPU systems using CUDA programming tools. We have introduced some novel schemes for dividing the projection data and image voxels over multiple GPUs to avoid runtime overhead and inter-device synchronization issues. We have also reduced the complexity of overlap calculation of the algorithm by eliminating the common projection plane and directly projecting the detector boundaries onto image voxel boundaries. To reduce the time required for calculating the overlap between the detector edges and image voxel boundaries, we have proposed a pre-accumulation technique to accumulate image intensities in perpendicular 2D image slabs (from a 3D image) before projection and after backprojection to ensure our DD kernels run faster in parallel GPU threads. For the implementation of our iterative MBIR technique we use a parallel multi-GPU version of the alternating minimization (AM) algorithm with penalized likelihood update. The time performance using our proposed reconstruction method with Siemens Sensation 16 patient scan data shows an average of 24 times speedup using a single TITAN X GPU and 74 times speedup using 3 TITAN X GPUs in parallel for combined projection and backprojection.* © *2017 Society for Imaging Science and*

INTRODUCTION

Model-Based Image Reconstruction (MBIR) algorithms provide the potential of producing quantitatively better images using data from conventional x-ray scanners than the linear reconstruction algorithms that are the industry standard. Conventional linear algorithms like the ones introduced by Feldkamp, David, and Kress (FDK) are mainly used in clinical settings for simplicity and low computation time.[1–3] Model-based image reconstruction algorithms may achieve the same image quality with lower dose[4,5] or better image quality at the same dose. Currently, patients go through multiple x-ray CT scans during image-guided radiation therapy, which elevates the potential risk for tissue damage and radiation-induced cancer.[6,7] Therefore, there is demand for fast iterative reconstruction algorithms that can produce higher quality images in clinically relevant time.

Model-based image reconstruction algorithms are typically iterative: the next image is computed based on the current image, a measure of error between the measured data and the data predicted from the current image, and a regularization function. Two important components of such algorithms are the forward data model and an algorithm for updating the image estimate based on errors in measurement space. For x-ray imaging, the forward data model is based on line integrals through attenuation images; we call the mapping from an image to a set of line integrals *forward projection*. We refer to the operator that is adjoint (or transpose) to forward projection as *backprojection*. Many MBIR algorithms use backprojection as a core component. In this paper, we describe implementations of forward projection and backprojection on a multi-GPU architecture that achieve significant speedup. We demonstrate the speedup for one choice of an MBIR algorithm, namely the alternating minimization (AM) algorithm.

▲ IS&T Member.

Received July 13, 2016; accepted for publication Sept. 29, 2016; published online Dec. 8, 2016. Associate Editor: Chunghui Kuo.

For MBIR algorithms to be feasibly implemented in practice, the computation time must be sufficiently low. The actual time demand depends on the application. In security applications, three-dimensional image volumes must be computed at the rate for bags to travel through the scanner. For many medical applications, the time depends on the availability of radiologists, which can vary widely. There are various pathways to decrease the time in iterative image reconstruction. One important pathway is through advanced algorithms from convex optimization theory.[23] A second pathway is through varying step sizes in existing algorithms, perhaps decreasing computation time for most images at the expense of not having guaranteed convergence properties. A third pathway is through parallel processing.

Many convex algorithms are designed to yield parallel updates that map well onto many computational architectures, after computing a forward projection and a backprojection. All such algorithms will have decreased computation time when coupled with efficient implementations of forward projection and backprojection.

One of the state-of-the-art projection algorithms, called distance driven (DD) projection and backprojection, was proposed by De Man and Basu.[8,9] In 2006, they proposed an extension to their algorithm called branch DD projection and backprojection in which they basically parallelized the inner loop of their overlap calculation.[10] They divided the overlap kernel in three distinct and independent steps: digital integration, interpolation, and digital differentiation. Schlifske et al.[11] proposed a 2D extension to the branchless DD algorithm in which they "pre-integrate" the 2D image slice of the image volume before projection and after backprojection. In our work, we use a similar method in which we pre-accumulate the image intensities in four perpendicular image slabs in a recursive manner before projection in order to accommodate the 3D helical nature of the data. We have also employed a recursive adjoint accumulation scheme after backprojection to retrieve our final 3D image volume. Our proposed method of pre-accumulation enables us to employ interpolation directly into the image accumulation array which reduces the computational burdens associated with the sequential integration of the original branchless DD method.

Along with the efforts to improve the structural aspects of reconstruction algorithms, there is also an overwhelming trend shifting toward multithreaded CPU and GPU implementations for improved time performance. The GPU technology has come a long way from its invention in the late 80s to its latest release of GeForce GTX TITAN X GPUs consisting of eight billion transistors on a single chip. Modern GPU technologies with their high memory bandwidth and peak arithmetic performance are rapidly outpacing their CPU counterparts.[12] Due to their inherent parallel architecture, GPUs can provide quite significant performance improvement for algorithms with highly pipelined structure. Current GPUs also provide very high global memory storage, which is ideal to fit the whole data volume and image array in the GPU itself during kernel execution, in turn eliminating the high latency penalty for accessing external memory. Due to all these advantages, it is quite logical to use GPUs to improve the speed of image reconstruction.

Over the years, several groups have accelerated their iterative reconstruction methods using GPUs. Andreyev et al.[13] have accelerated their blob-based iterative reconstruction using a Tesla GPU. Jia et al.[14] implemented a low dose cone beam CT reconstruction with total variation regularization on an NVIDIA Tesla C1060 GPU. McGaffin et al.[15] proposed a multi-GPU based fast converging stochastic group ascent algorithm to perform dual maximization and implemented their algorithm on NVIDIA Tesla C2050 GPUs. Wu et al.[16] accelerated separable footprint based projection and backprojection algorithms using NVIDIA Tesla C2050 GPUs. Quivira et al.[17] developed an iterative 3D reconstruction algorithm for sparse x-ray CT data on Titan X GPUs.

In our work we describe a Poisson model for the measured x-ray CT data. The maximum likelihood estimation problem is then reformulated as the double minimization of an I-divergence problem. An AM algorithm is then formulated with the addition of a Huber type penalty function. After that we focus on the parallelization of the branchless DD projection and backprojection over multiple GPUs. We first simplify the overlap computation of branchless DD algorithm by projecting detector boundaries directly onto the image voxel boundaries. After that, we added a pre-accumulation scheme, which reduces the sequential integration burden on individual GPU threads. Next, we present a pseudocode for the implementation of our proposed algorithm on single and multiple GPUs. Lastly we have validated our overall parallelization scheme by reconstructing images from Siemens Sensation 16 helical CT data using the AM algorithm and its ordered subsets version.

METHODS

Mathematical Model

Multislice helical x-ray CT is a useful imaging modality in many clinical applications and is now in widespread use. This type of CT is inherently 3D because the x-ray tube continuously projects a cone beam through the patient as the patient is translated through the scanner. Each detector row captures data in a partial rotation of the gantry that corresponds to each image slice.

In this paper, we consider a mono-energetic scatter-free statistical model to account for the x-ray photon randomness as was done previously.[18,19] At the basis of our statistical model, we assume the photons arrive at the detectors in accordance with a photon counting process. Let the 3D image volume of linear attenuation coefficients (in mm^{-1}) be represented by the vector μ. Let i denote a ray path between the x-ray source and a pixel in the multi-row detector array and j denote a voxel in the image volume. The measured transmission data, d, is modeled as originating from independent Poisson counting processes. In discretized

form, the mean value of d_i is

$$g_i(\mu) = I_i e^{-l_i(\mu)}, \tag{1}$$

where $l_i(\mu)$ is the forward projection given by

$$l_i(\mu) = \sum_j \boldsymbol{a}_{ij}\mu_j. \tag{2}$$

I_i is the mean number of counts in the absence of an attenuating medium, and μ_j is the linear attenuation coefficient in voxel j. The system matrix elements $\boldsymbol{a}_{ij}$ comprise the appropriately discretized point spread function relating the projection space to the image space. If projection i does not pass through voxel j, then $\boldsymbol{a}_{ij}$ is zero.

Alternating Minimization Reconstruction Algorithm

For our AM algorithm we use the maximum likelihood solution derived by O'Sullivan and Benac.[19] The problem was formulated as the double minimization of an I-divergence over a linear and exponential family, thereby resulting in a closed-form update for each iteration. The objective function to be minimized for the mono-energetic case is

$$I[d||g(\mu)] \triangleq \sum_i [d_i \ln(d_i/g_i(\mu)) + g_i(\mu) - d_i]. \tag{3}$$

For our implementation of the AM algorithm, we compute two backprojections using a branchless DD algorithm on measured data and predicted data which are represented below as b_j and $\hat{b}_j^{(k)}$

$$b_j \triangleq \sum_i \boldsymbol{a}_{ij} d_i \tag{4}$$

$$\hat{b}_j^{(k)} \triangleq \sum_i \boldsymbol{a}_{ij} g_i(\hat{\mu}^{(k)}). \tag{5}$$

From O'Sullivan and Benac,[19] the update for linear attenuation coefficients is

$$\hat{\mu}_j^{(k+1)} \triangleq \left[\hat{\mu}_j^{(k)} - \frac{1}{Z}\ln\left(\frac{b_j}{\hat{b}_j^{(k)}}\right)\right], \tag{6}$$

where $\hat{\mu}^k$ is the estimate of μ at iteration k and Z is an auxiliary variable that satisfies $Z = \max_i \sum_j \boldsymbol{a}_{ij}$.

Since the measured data is noisy, it is necessary to regularize the optimization problem to prevent the algorithm from over-fitting the data through unrealistic images. We take an approach analogous to that of Erdögan et al.[20] and decouple the image variables of our penalized objective function such that all the voxels can still be updated in parallel. To derive the algorithm for penalized maximum likelihood estimation, we add a penalty term, $R(\mu)$, to the objective function used in the AM reconstruction, and weight it by a regularization parameter λ, where λ is a scalar that reflects the amount of smoothing desired. A larger value will give emphasis to the penalty term (i.e., the prior expectation that the image will be smooth), whereas a smaller value will give more emphasis to the I-divergence term (i.e., the discrepancy between the measured data and the data estimated by the model). The added penalty term is defined as

$$R(\mu) \triangleq \sum_j \sum_{j' \in N_j} \omega_{j,j'} \psi(\mu_j - \mu_{j'}). \tag{7}$$

For 3D regularization, we use the 26-voxel neighborhood N_j surrounding voxel j. The weights $\omega_{j,j'}$ control the relative contribution of each neighbor. The potential function $\psi(t)$ is a symmetric convex function that penalizes the difference between the values of neighboring voxels. For computational simplicity, we use a modified potential function used by Lange,[21]

$$\psi(t) \triangleq \delta^2\left[\left|\frac{t}{\delta}\right| - \ln\left(1 + \left|\frac{t}{\delta}\right|\right)\right], \tag{8}$$

where δ is a parameter that controls the transition between a quadratic region (for smaller t) and a linear region (for larger t). For our specific reconstruction, the first and last set of image slices are not included in the penalty calculation because those slices will have severe artifacts due to cone beam truncation. Calculating the penalty for those slices could negatively impact reconstruction of the inner slices since the artifacts do not have the type of structure that can meaningfully be penalized by $R(\mu)$. The overall problem is then to find the penalized likelihood estimate,

$$\hat{\mu}_j^{(k+1)} = \text{argmin}_{\hat{\mu}_j^{(k)} \geq 0} I[d||g(\hat{\mu}_j^{(k)})] + \lambda R(\hat{\mu}_j^{(k)}). \tag{9}$$

For our specific implementation, we use Newton's method to find the optimum iteratively. It is worth noting that our solution in (6) is a special case of (9) when $\lambda = 0$. The complete AM algorithm scheme is shown in Figure 1.

Branchless Distance Driven Operators

The core calculation of the algorithm is the computation of the overlap between the projection of an individual slab of the image volume onto a 2D detector array. For our specific reconstruction, we used helical CT geometry. We have also exploited the quarter rotation symmetry[22] because it significantly reduces our computational burden. In our algorithm, the overlap calculations are performed directly at the level of the slab of interest. This differs slightly from the method proposed by De Man and Basu,[9] where the overlap calculations are performed in the xz or yz plane passing through the origin. In that case, both the flattened voxel edges and detector edges would need to be projected onto the plane passing through the origin. In our implementation, the only projection calculations are from the detector edges to the slab. The coordinates of the source-to-detector ray intersections with the flattened slab determine the 2D rectangular region of the slab that contributes to each detector element. These rays are constructed using the edges of each detector element. For the completion of an x-ray projection image for a particular view angle, all the slab contributions are aggregated for a particular detector array. The contribution is also scaled by the length of the intersection of the ray through that slab. For our particular

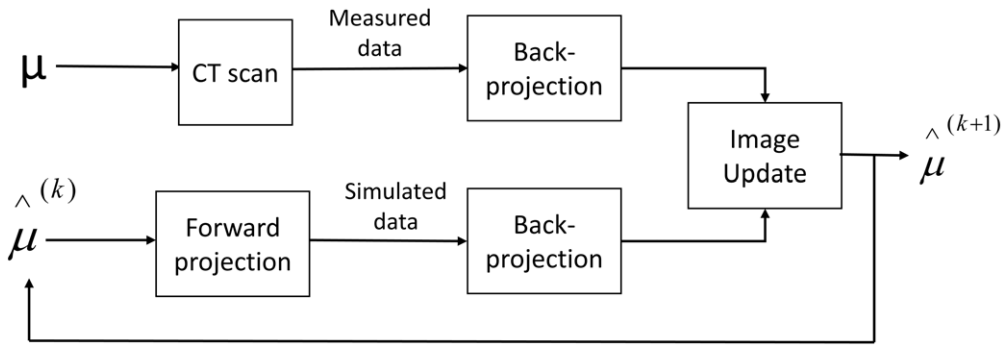


Figure 1. Schematic diagram for iterative AM reconstruction algorithm.

reconstruction, we assumed the slabs are flat and of uniform thickness.

Parallel Implementation of Branchless Distance Driven Forward Projection

First, we consider the contribution from a 1D pixel array (i.e., one slab of a 2D image) to a detector element at a fixed view angle. The pixels are uniformly spaced and represent a continuous function, $f(x)$, using a rectangle basis of unit width,[10]

$$f(x) \triangleq \sum_i f_i \phi(x-i), \tag{10}$$

where

$$\phi(x) = \begin{cases} 0 & x < -0.5 \\ 1 & -0.5 \le x \le 0.5 \\ 0 & x > 0.5. \end{cases} \tag{11}$$

We wish to find the total contribution of the pixel array to detector element k with edges $x = u_1$ and $x = u_2$. This is mathematically expressed as

$$\begin{aligned} g_k &\triangleq \frac{1}{u_2-u_1}\int_{u_1}^{u_2} f(x)\,\mathrm{d}x \\ &= \frac{F(u_2)-F(u_1)}{u_2-u_1}, \end{aligned} \tag{12}$$

where

$$F(u) \triangleq \int_{-\infty}^{u} f(x)\,\mathrm{d}x. \tag{13}$$

Let $K \triangleq \lfloor u \rfloor$, i.e., floor (u). Plugging it into (10), Eq. (13) can be rewritten as

$$F(u) = \sum_i f_i \int_{-\infty}^{u} \phi(x-i)\,\mathrm{d}x \tag{14}$$

$$= \sum_{i=0}^{K-1} f_i \int_{-\infty}^{K} \phi(x-i)\,\mathrm{d}x + f_k \int_K^u \phi(x-K)\,\mathrm{d}x \tag{15}$$

$$= \sum_{i=0}^{K-1} f_i + (u-K) f_K. \tag{16}$$

Next, we can define an accumulated pixel array,

$$A[m] \triangleq \sum_{i=0}^{m-1} f_i. \tag{17}$$

We can rewrite Eq. (16) using (17) as follows:

$$F(u) = A[K] + (u-K) f_K \tag{18}$$

$$F(u) = A[K] + (u-K)(A[k+1] - A[K]). \tag{19}$$

Now $F(u)$ can be calculated simply in terms of the pre-accumulated array A, and the original pixel values f_i are no longer needed. In fact, (19) is a linear interpolation into array A. The final step to calculate g_k is to perform the operation in (12)

$$g_k \triangleq \frac{1}{(u_2-u_1)(v_2-v_1)} \int_{u_1}^{u_2}\int_{v_1}^{v_2} f(x,z)\,\mathrm{d}z\,\mathrm{d}x. \tag{20}$$

We can define a continuous-coordinate slab using separable rectangular functions

$$f(x,z) \triangleq \sum_i \sum_j f_{ij} \phi(x-i)\phi(z-j). \tag{21}$$

We can represent in-plane calculations for each basis position j in the z direction

$$F_j(u) = A_j[K] + (u-K)(A_j[K+1] - A_j[K]), \tag{22}$$

where

$$A_j[m] \triangleq \sum_{i=0}^{m-1} f_{ij}. \tag{23}$$

This leads to

$$\begin{aligned} g_k &= \frac{1}{(u_2-u_1)(v_2-v_1)} \\ &\quad \times \sum_j F_j(u_2) - F_j(u_1) \int_{v_1}^{v_2} \phi(z-j)\,\mathrm{d}z \end{aligned} \tag{24}$$

$$= \frac{G(u_1,u_2,v_2) - G(u_1,u_2,v_1)}{(u_2-u_1)(v_2-v_1)}, \tag{25}$$

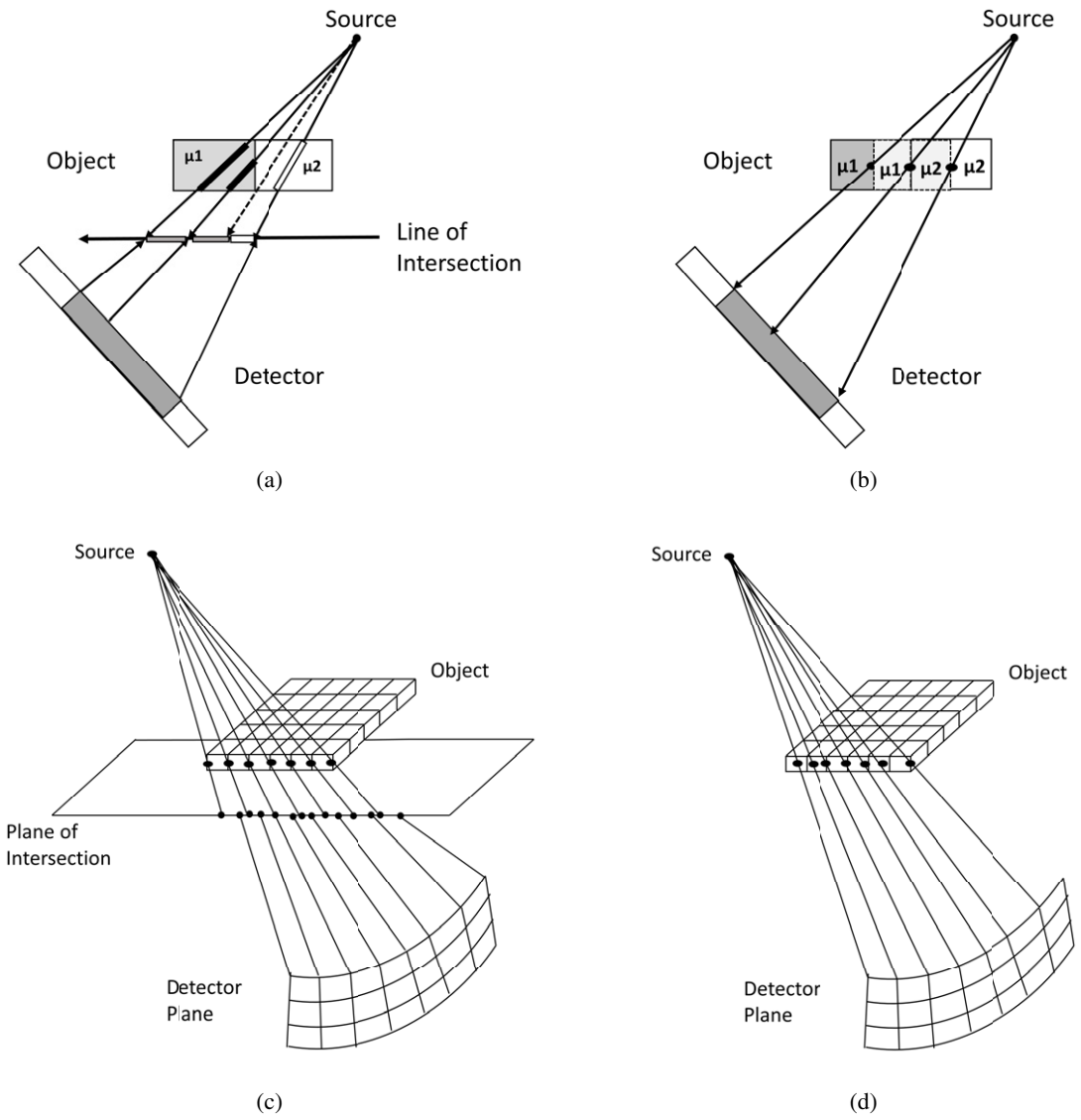


Figure 2. (a) Schematic representation of De Man and Basu's[9] 2D distance driven method. (b) Schematic representation of our 2D distance driven method. (c) Schematic representation of De Man and Basu's[9] 3D distance driven method. (d) Schematic representation of our 3D distance driven method.

where

$$G(u_1, u_2, v) = \sum_j F_j(u_2) - F_j(u_1) \int_{-\infty}^{v} \phi(z - j)\, \mathrm{d}z. \quad (26)$$

Similarly, we can define an accumulated voxel array in the z direction

$$C_{u_1,u_2}[n] \triangleq \sum_{j=0}^{n-1} B_j(u_1, u_2). \quad (27)$$

Analogous to (19) we define $J \triangleq \lfloor v \rfloor$. We can write

$$G(u_1, u_2, v) = C_{u_1,u_2}[J] + (v - J)(C_{u_1,u_2}[J+1] - C_{u_1,u_2}[J]). \quad (28)$$

We can also write $\sum_j F_j(u_2) - F_j(u_1)$ as weighted sum of few elements of $A_j[m]$,

$$B_j(u_1, u_2) = \sum_m \omega_m A_j[m], \quad (29)$$

where ω_m is nonzero for up to four distinct values of m, as determined by (19) and (22). Therefore, the slab can be pre-accumulated in both the x and z directions, as shown below:

$$C_{u_1,u_2}[n] = \sum_{j=0}^{n-1} \sum_m \omega_m A_j[m] \quad (30)$$

$$= \sum_m \omega_m \sum_{j=0}^{n-1} \sum_{i=0}^{m-1} f_{i,j} \quad (31)$$

$$= \sum_m \omega_m S[m, n], \quad (32)$$

where

$$S[m, n] \triangleq \sum_{j=0}^{n-1} \sum_{i=0}^{m-1} f_{i,j}. \quad (33)$$

Finally, this accumulation can be written in recursive form for faster calculation as follows:

$$S[m, n] = \sum_{j=0}^{n-1} A_j[m] \tag{34}$$

$$= \sum_{j=0}^{n-2} A_j[m] + A_{n-1}[m] \tag{35}$$

$$= S[m, n-1] + \sum_{i=0}^{m-1} f_{i,n-1}. \tag{36}$$

For the projection model, as shown above, we pre-accumulate original pixel values in a recursive manner to a pre-accumulation array corresponding to four perpendicular slabs, each contributing to a different orientation of our view angle. After the pre-accumulation, the original voxel values are no longer required. In fact, we perform direct interpolation of detector edges onto this accumulation array which gives us a big boost on the time performance over the sequential computation of digital integration for every overlap computation. Before performing interpolation and differentiation, we determine which part of the algorithm could be divided into independent processes to run on a single GPU thread. The way branchless projection methods are structured, the interpolation and digital differentiation for each slab at each quarter rotation are independent of one another, so it can be implemented on a single GPU thread.

Parallel Implementation of Branchless Distance Driven Backprojection

Backprojection for the DD kernel is defined as the transpose of the forward projection operator. Using flow graph reversal, the transpose of the entire kernel can be done by transposing each sub-operation and performing them in the reverse order, i.e.:

(a) Transposed digital differentiation,

(b) Transposed linear interpolation or "anterpolation,"

(c) Transposed integration.

By writing out the 2D slab accumulation operation (34) in matrix form, it can be shown that the transpose of slab accumulation is

$$f^*_{i,j} = \sum_{n=j+1}^{N_z} \sum_{m=i+1}^{N_x} S[m, n], \tag{37}$$

where N_x and N_z are the number of voxels in the two directions, respectively. This operation can also be written recursively for faster calculation. If we let

$$D[i, n] \triangleq \sum_{m=i+1}^{N_x} S[m, n], \tag{38}$$

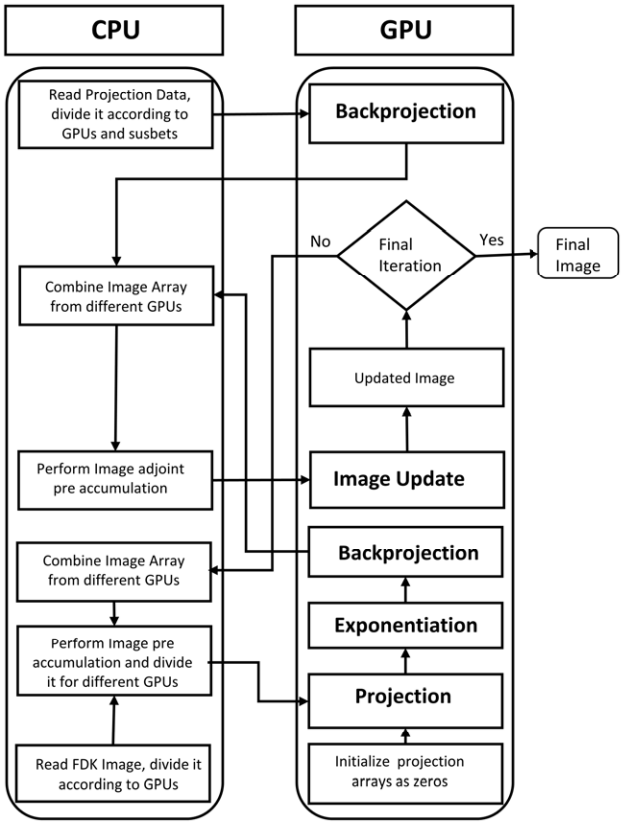


Figure 3. Schematic diagram of the GPU implementation of the iterative reconstruction algorithm.

then

$$f^*_{i,j} = \sum_{n=j+1}^{N_z} D[i, n] \tag{39}$$

$$= \sum_{n=j+2}^{N_z} D[i, n] + D[i, j+1] \tag{40}$$

$$= f^*_{i,j+1} + D[i, j+1] \tag{41}$$

$$= f^*_{i,j+1} + \sum_{m=i+1}^{N_x} S[m, j+1]. \tag{42}$$

For transposed digital integration we perform the similar recursive post-accumulation technique over the accumulated backprojection array to retrieve the individual voxel values from four mutually perpendicular image slabs.

Implementation on Multiple GPUs

Each GPU is assigned a contiguous group of projections whose cardinality is a multiple of the number of views in a quarter rotation. With ordered subsets (OS), each subset consists of evenly distributed projections over all GPUs. For example, if two subsets are used, subset 0 would consist of the even-indexed source angles on each processor, and subset 1 would consist of the odd-indexed source angles.

This design allows for theoretically perfect load balancing (in the absence of memory-related latencies) during

```
Perform 2D accumulation of μ for each quarter rotation according to equation (36)
Number of GPU threads launched = Number of views within 1st quarter rotation * number of slabs in accumulated
image * number of quarter rotations assigned to each GPU
Launch GPU kernel
for all GPU blocks in parallel do
  for all threads in a block do
    begin GPU thread calculation
    for every detector column
      determine if the channel contribution to slab is nonzero
      interpolate slab at detector column edge
      differentiate the value of the interpolation
      for every detector row
        interpolate column differentiation results at detector row edge
        differentiate row interpolation values at row edges
        accumulate the differentiation value to the corresponding element in projection array
      end for
    end for
  end of GPU thread calculation
  end for
end for
weight projection by lengths of intersection through the slab
end kernel
```

A basic pseudocode of the 3D implementation of our proposed forward projection algorithm

forward and backprojection since each GPU essentially makes use of the same number of nonzero $\boldsymbol{a}_{ij}$ elements. The full-sized accumulation images and the projection data corresponding to each subset are stored in GPU global memory.

In our approach, we systematically add slices with minimal synchronization overhead between the devices. We have also determined the maximum block size that can be summed concurrently by all devices.

Forward projection is straightforward in terms of global memory access, since each device stores values in separate portions of the projection data array, and access to the accumulation image is read only. However, if we were to perform backprojection directly into the full-sized accumulation images, we would have serious memory contention issues since multiple devices would be writing to the same array elements simultaneously. Instead, each device performs backprojection to its own private accumulation image arrays (of reduced size compared to the full-sized arrays). This eliminates any need for synchronization during the backprojection of a device's set of views. Once each device is done backprojecting its set of views, the partial accumulation image arrays are summed into the full-sized accumulation image arrays. Fig. 3 illustrates the process by which non-overlapping groups of slices from each partial array can be added simultaneously without memory contention. After each block, a barrier synchronization construct is used to ensure each device has finished summing the current block of slices to the full-sized arrays.

However, these two approaches create the following constraints on several parameters as follows:

- Total number of views must be a multiple of the number of views in one quarter rotation.
- Total number of quarter rotations must be a multiple of the number of GPU devices.
- The number of subsets must divide into the number of views per quarter rotation evenly.

For measured data where these constraints were not satisfied, we pad the measured sinograms with zeros to increase the number of views.

To minimize the overhead time that occurs in data copying, kernel launch, etc., we create the same number of

```
Number of GPU threads launched = Number of views within 1st quarter rotation * number of slabs in accumulated
image * number of quarter rotations assigned to each GPU
launch GPU kernel
for all GPU blocks in parallel do
   for all threads in a block do
      begin GPU thread calculation
      weight projection by lengths of intersection through slab
       for each detector column
          determine if the channel contribution to slab is nonzero
          for every detector row
              adjoint differentiate the corresponding element in projection array in the row direction
              anterpolate results for corresponding row edge
          end for
         anterpolate results for last row edge
         adjoint differentiate for corresponding detector column edge for all relevant column edges
         anterpolate result for corresponding detector column edge to slab
      end for
      anterpolate result for last detector column edge to slab
    end of GPU thread calculation
 end for
end for
end kernel
perform 2D adjoint accumulation for every quarter according to Eq. (42)
sum the four adjoint accumulation images into μ
```

A basic pseudocode of the 3D implementation of our proposed backprojection algorithm

CPU threads as the number of GPUs to be utilized. Each of the threads interacts with an individual GPU. Each of them copies input data from the CPU to the GPU, executes the kernel, and copies results back to the CPU. The host CPU waits for all GPU devices to complete and merges results into one.

RESULTS

To compare both time performance and image quality, we start with an Intel Core i7 5960x with 8 cores, 16 threads, clocked at 3 GHz, with 20 MB cache and 64 GB of memory. For our GPU implementation, we used GeForce GTX TITAN X. TITAN X is based on Maxwell architecture with 3072 CUDA cores and 24 streaming multiprocessors (SMs) running at 1.2 GHz. Each block contains 65536 registers and 48 KiB of shared memory. Some of the highlights of TITAN X hardware are shown in Table I.

Table I. Hardware specification of TITAN X.

Single precision	7.468 TeraFLOP/s
Double precision	233.376 GigaFLOP/s
Multiprocessors	24
Clock rate	1.216 GHz
Global Memory bandwidth	336.48 GB/s
L2 Cache size	3MiB
CUDA cores	3072
Shared memory per block	48KiB

We used raw sinogram data from a Siemens Sensation 16. The parameters of the measured data and reconstructed images are shown in Table II:

Figures 5 and 6 show resulting image reconstruction from running 10 iterations with 145 ordered subsets

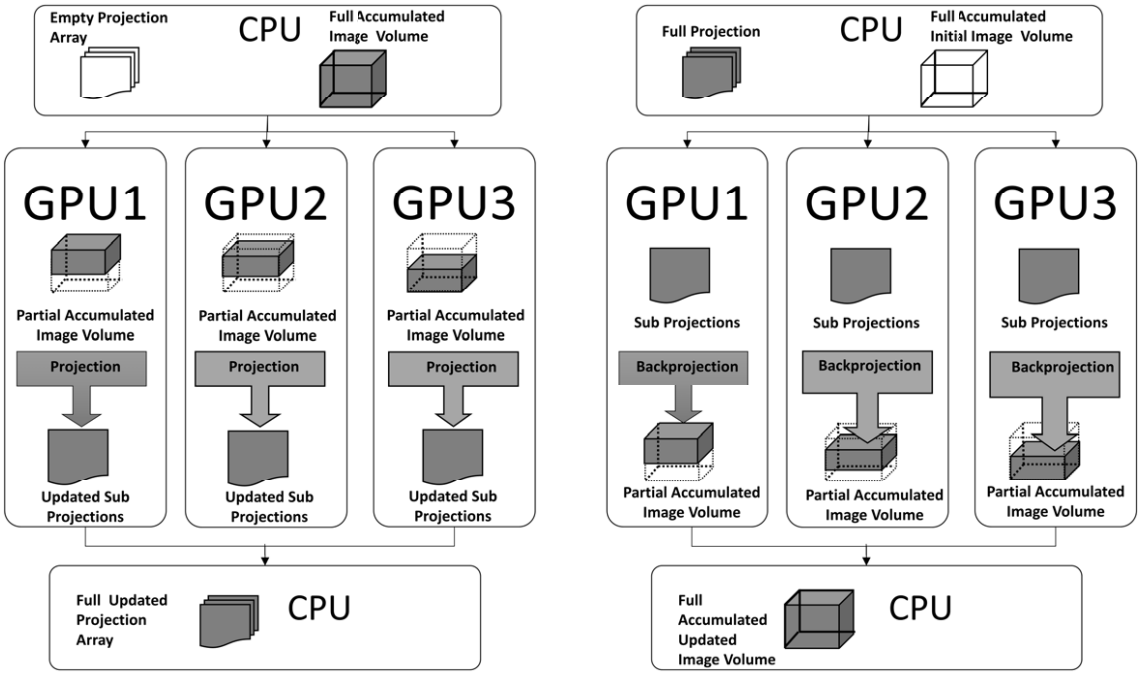


Figure 4. (a) Schematic representation of Multi-GPU implementation of branchless DD projection. (b) Schematic representation of Multi-GPU implementation of branchless DD backprojection.

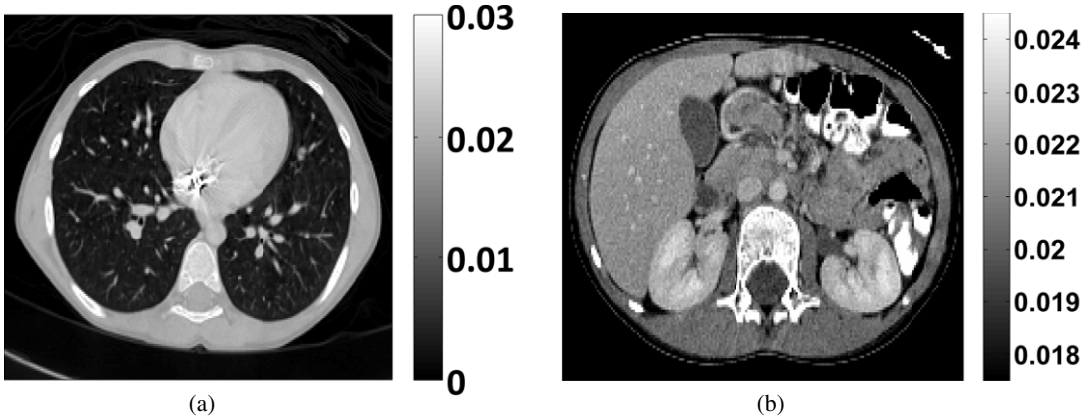


Figure 5. Axial Slices of 3D AM Reconstruction of linear attenuation coefficients (in mm^{-1}) of (a) abdomen and (b) lung after 10 iterations of 145 ordered subsets.

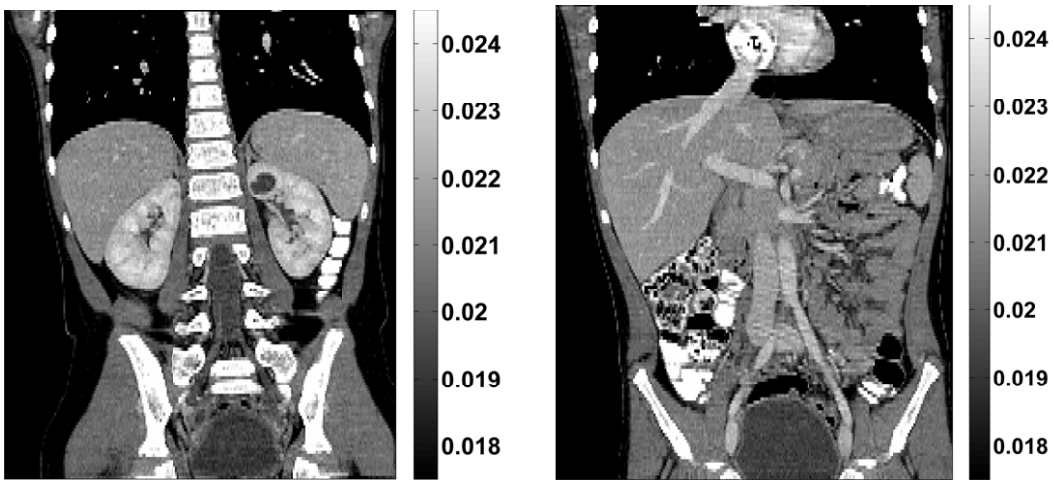
Table II. Parameters of measured data and image.

No. of views	13920
No. of detector channels	672
No. of detector rows	16
No. of image slices	164
No. of pixels/slice	512 × 512

using alternating minimization update with Huber type log likelihood penalty on 3GPUs. Fig. 6 shows time performance results from the implementation of our algorithm without any ordered subsets. The leftmost bar in Figure 7 is the execution time of the baseline serial version and the remaining bars

Table III. Execution times by using different CPU and GPU configurations for single branchless DD forward and backprojection.

Operations	Execution Time (seconds)			
	Single threaded CPU	16 threaded CPU	Single GPU	Multi-GPU
Pre-accumulation	8.1	1.7	0.570	0.21
Projection	433	92	15	4.7
Exponentiation	1.1	0.25	0.07	0.029
Backprojection	435	95	22	7.6
Image Update	4.8	1.2	0.17	0.06
Total	882	190.15	37.81	12.6



(a) (b)

Figure 6. Coronal Slices of 3D AM Reconstruction of linear attenuation coefficients (in mm^{-1}) of abdomen after 10 iterations of 145 ordered subsets.

show runtimes for the specific optimizations using multiple CPU threads and multiple GPU devices. Table III shows the time of execution of each component of our algorithm with different hardware configurations. For the baseline serial version, we run our projector algorithms on a single CPU core with nested for loops representing the parallel GPU threads. For multithreaded CPU implementation, each CPU core launches two hyper threads for every logical processor in the core. Each hyper thread basically acts as a standalone GPU device. Instead of parallel GPU threads, we use a corresponding number of nested for loops. We also use a barrier synchronization to wait for every CPU thread to finish its projection and backprojection in their private projection and image accumulation arrays respectively. To calculate the parallelization efficiency of the multithreaded CPU version we define our speedup ratio according to Amdahl's law as follows

$$S = \frac{T_1}{T_N} < \frac{1}{\left(f + \frac{1-f}{N}\right)} < \frac{1}{f} \quad \text{as } N \to \infty, \tag{43}$$

where, T_1 and T_N are elapsed times of 1 and N workers. f is the fraction of the code that is not parallelizable. The parallel efficiency is then defined as,

$$E = S/N. \tag{44}$$

From our experimentation with $N = 16$ CPU threads, we get $S = T_1/T_N = 4.7$ for the projection operation. As a result, $f = 0.1603$ and parallel efficiency is $E = 0.2963$. So we can conclude, our multithreaded CPU implementation can achieve a maximum speedup of 6.2 times for the projection operation.

Since we can divide the projection array according to its number of ordered subsets and the number of GPU devices available, the effective size of the projection array passed to GPUs is much smaller than the size of the partial image accumulation array. As a result, the backprojection

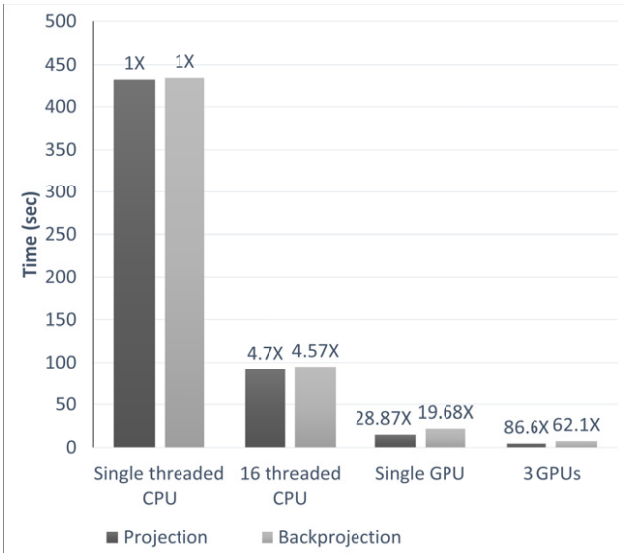


Figure 7. Time performance improvement using different CPU and GPU configurations for single Branchless DD projection and backprojection.

operator tries to accumulate and write the result on a much bigger image accumulation array than the projection array from which it tries to read. So the time required for backprojection is higher than for projection. The difference is much more significant when we use more ordered subsets since the number of subsets only reduces the volume of projection array keeping the size of partial accumulation array unchanged.

Figure 8 shows the time required for single iteration of different ordered subset configurations by using three GPUs in parallel. The time needed to combine partial image accumulation arrays from different GPU devices after every backprojection increases the iteration time for ordered subset configurations. For ordered subset implementation, we also need to perform measured data backprojection after every subset iteration since the measured data backprojection array for all the subsets cannot be saved in finite device memory. In Figure 9, we show the change in objective function

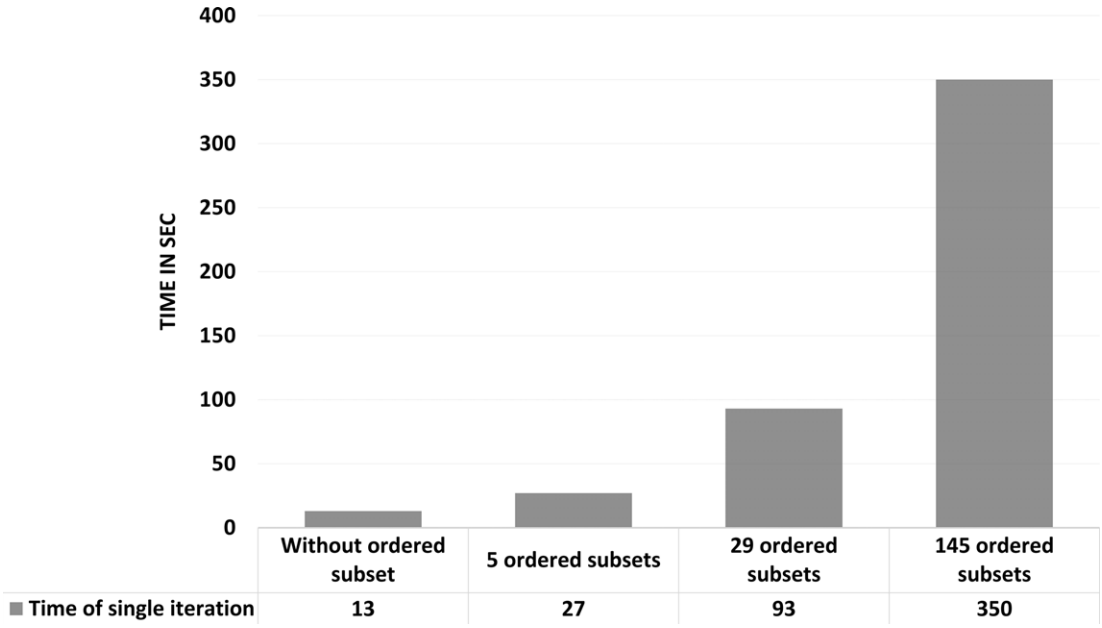


Figure 8. Time performance for a single iteration of various ordered subsets.

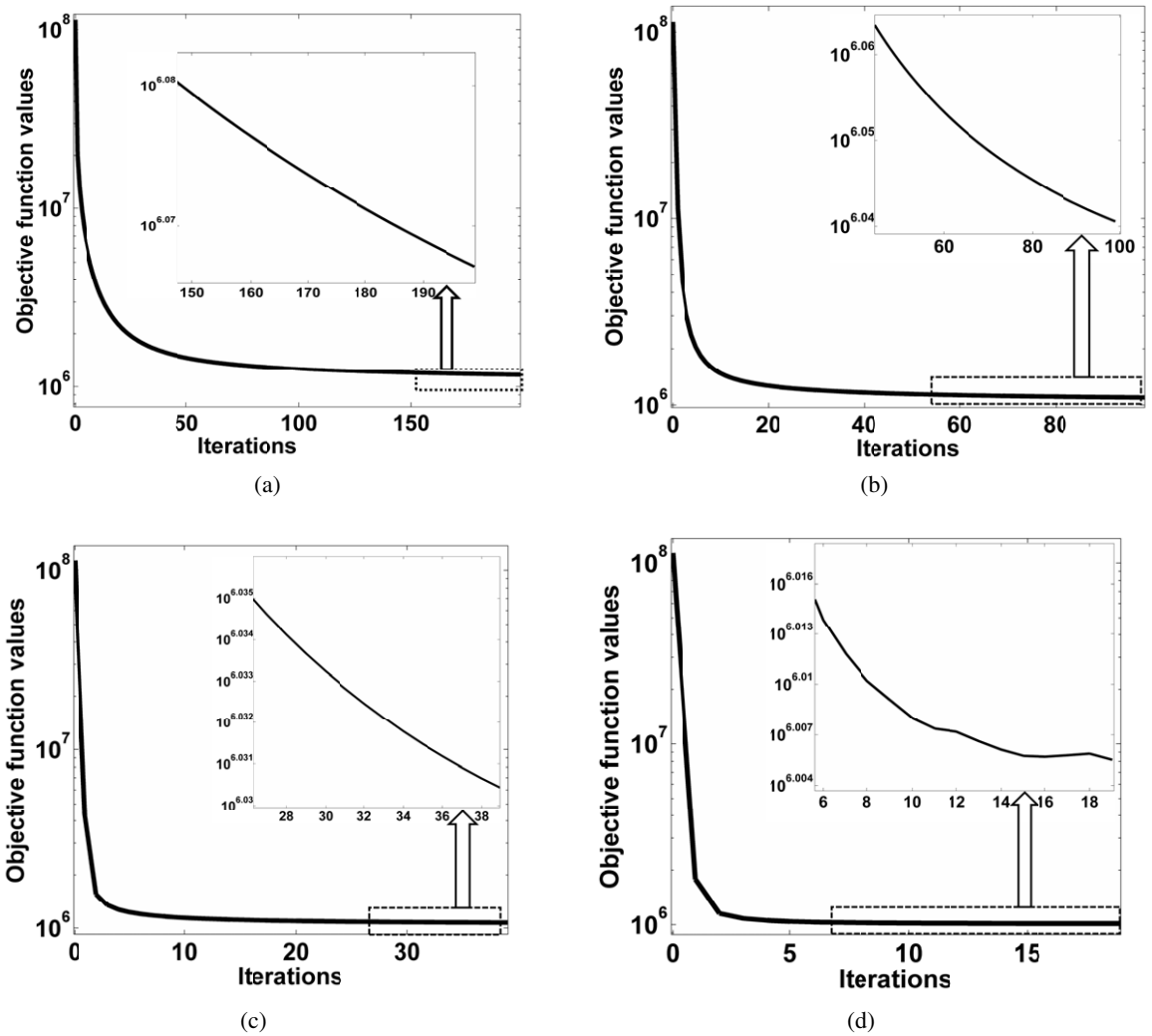


Figure 9. Objective function values versus iteration number (a) without ordered subsets, (b) 5 ordered subsets, (c) 29 ordered subsets and (d) 145 ordered subsets.

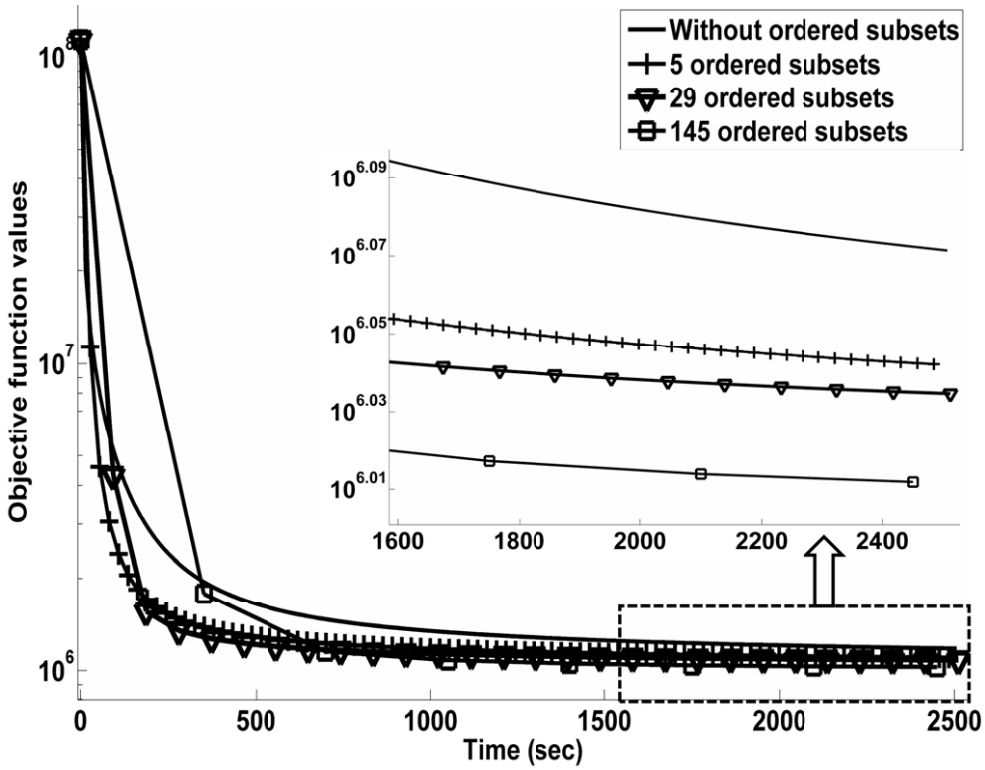


Figure 10. Objective function values in different time interval for various ordered subset configurations.

values (defined in equation (9)) with iteration number for various ordered subset configurations. Since minimizing the objective function values will maximize the penalized log likelihood between the measured data and our estimated data by the model, we can use this distance method to estimate the accuracy and noise reduction of our reconstruction. In Figure 10, we show the change in objective function values with corresponding time interval for different ordered subset configurations. The objective function values at 0th iteration of Fig. 9 and 0th second of Fig. 10 denote the value of objective function between measured data and projection sinogram of FDK reconstruction of the data. The significant decrease in the objective function values clearly illustrates the improvement in image quality with our proposed reconstruction algorithm. In the end, we can clearly conclude that our optimizations are effective and that our multi-GPU approach is beneficial for both forward and backprojection cases.

CONCLUSIONS

We have observed that our approach of using multiple GPUs to reconstruct images gives us better performance in computational cost compared to our best available CPU configuration. Our primary contribution is a novel approach to pre-accumulate for projection (see equation (36)) and adjoint pre-accumulate for backprojection (see equation (42)) in three-dimensional branchless DD algorithm. Elimination of the common projection for 3D branchless DD projectors reduces complexity of the algorithm (see Fig. 2). We can also observe that computational time shows a linear decrease in time performance with the addition of more GPUs. Use of texture memory of the GPU devices for storing our accumulation array is expected to reduce our computation time of backprojection. We can expect to reduce run times with more GPUs (see Fig. 7), which opens the door to exciting new possibilities in clinical settings. For precision critical applications we can use the double precision floating point with TITAN Z GPUs, with some performance degradation compared to our single precision TITAN X GPUs.

ACKNOWLEDGMENT

This work was supported in part from grants (No. R01 EB019135, B. Whiting, PI; No. R01 CA149305, J. Williamson, PI) awarded by the National Institutes of Health.

REFERENCES

[1] D. A. Jaffray and J. H. Siewerdsen, "Cone-beam computed tomography with a flat-panel imager: initial performance characterization," Med. Phys. **27**, 1311–1323 (2000).
[2] L. A. Feldkamp, L. C. Davis, and J. W. Kress, "Practical cone beam algorithm," J. Opt. Soc. Am. A **1**, 612–619 (1984).
[3] J. L. Jianchun, C. Papachristou, and R. Shekhar, "An FPGA-based computing platform for real-time 3D medical imaging and its application to cone-beam CT reconstruction," J. Imaging Sci. Technol. **49**, 237–245(9) (2005).
[4] A. K. Hara, R. G. Paden, A. C. Silva, J. L. Kujak, H. J. Lawder, and W. Pavlicek, "Iterative reconstruction technique for reducing body radiation dose at CT: feasibility study," Am. J. Roentgenol. **193**, 764–771.
[5] P. Prakash, M. K. Kalra, A. K. Kambadakone, H. Pien, J. Hsieh, M. A. Blake, and D. V. Sahani, "Reducing abdominal CT radiation dose with adaptive statistical iterative reconstruction technique," Investigative Radiol. **45**, 202–210.

[6] ICRP, "Managing patient dose in computed tomography" *Ann. ICRP* **30**, 7–45 (2000).
[7] M. W. Kan, L. H. Leung, W. Wong, and N. Lam, "Radiation dose from cone beam computed tomography for image-guided radiation therapy," Int. J. Radiat. Oncol. Biol. Phys. **70**, 272–279 (2008).
[8] B. De Man and S. Basu, "Distance-driven projection and backprojection," *Nuclear Science Symposium Conf. Record, 2002 IEEE* (IEEE, Piscataway, NJ, 2002), Vol. 3.
[9] B. De Man and S. Basu, "Distance-driven projection and backprojection in three dimensions," Phys. Med. Biol. **49**, 2463–2475 (2004).
[10] S. Basu and B. De Man, "Branchless distance driven projection and backprojection," Proc. SPIE **6065**, 6065Y (2006).
[11] D. Schlifske and H. Medeiros, "A fast GPU-based approach to branchless distance-driven projection and back-projection in cone beam CT," Proc. SPIE **9783**, 97832W (2016).
[12] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU Computing," Proc. IEEE **96**, 879–899 (2008).
[13] A. Andreyev, A. Sitek, and A. Celler, "Acceleration of blob-based iterative reconstruction algorithm using Tesla GPU," *2009 IEEE Nuclear Science Symposium Conf. Record (NSS/MIC)* (IEEE, Piscataway, NJ, 2009), pp. 4095–4098.
[14] X. Jia, Y. Lou, J. Lewis, R. Li, X. Gu, C. Men, W. Y. Song, and S. B. Jiang, "GPU-based fast low-dose cone beam CT reconstruction via total variation," J. XRay Sci. Tech. **19**, 139–154 (2011).
[15] M. G. McGaffin and J. A. Fessler, "Alternating Dual Updates Algorithm for X-ray CT Reconstruction on the GPU," IEEE Trans. Comput. Imaging **1**, 186–199 (2015).
[16] M. Wu and J. A. Fessler, "GPU acceleration of 3D forward and backward projection using separable footprints for X-ray CT image reconstruction," *Proc. Int'l. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med.* (Postdam, Germany, 2011), vol. 6, p. 021911.
[17] F. Quivira, S. Bedford, R. Moore, J. Beaty, and D. Castañón, "Sparse data 3-D X-ray reconstructions on GPU processors," *IS&T Electronic Imaging: Computational Imaging XIV* (IS&T, Springfield, VA, 2016), pp. 1–5.
[18] J. A. Fessler, "Statistical image reconstruction methods for transmission tomography," in *Handbook of Medical Imaging: Medical Image Processing and Analysis*, edited by M. Sonka and J. M. Fitzpatrick (SPIE, Bellingham, WA, 2000), Vol. 2, pp. 1–70.
[19] J. A. O'Sullivan and J. Benac, "Alternating minimization algorithms for transmission tomography," IEEE Trans. Med. Imaging **26**, 283–297 (2007).
[20] H. Erdögan and J. A. Fessler, "Ordered subsets algorithms for transmission tomography," Phys. Med. Biol. **44**, 2835–2851 (1999).
[21] K. Lange, "Convergence of EM image reconstruction algorithms with Gibbs smoothing," IEEE Trans. Med. Imaging **9**, 439–446 (1990).
[22] D. Keesing, "*Development and Implementation of Fully 3D Statistical Image Reconstruction Algorithms for Helical CT and HalfRing PET Insert System*," All theses and dissertations (ETDs), Paper 427, pp. 48–50 (2009).
[23] Q. Xu, D. Yang, J. Tan, A. Sawatzky, and M. A. Anastasio, "Accelerated fast iterative shrinkage thresholding algorithms for sparsity-regularized cone-beam CT image reconstruction," Medical Physics **43**, 1849–1872 (2016).